ELECTRIC VEHICLE ROUTING: AN ATTEMPT

Federico Rafael Jaramillo
Universidad EAFIT
Colombia
fjaram18@eafit.edu.co

Mauricio Toro
Universidad EAFIT
Colombia
mtorobe@eafit.edu.co

ABSTRACT

The trend towards renewable energy has led to a need of optimization in the routing and electric consumption of electric cars in order to make it more efficient. Since the fossil fuels used in common vehicles damages the environment, electric cars have become a stronger alternative for transportation. Since it is still a newer technology, efficient ways to route and mobilize electric vehicles are of utmost importance in order to make them a viable option against the standard more common fossil fueled vehicles. A lot of research has been done on the subject in recent years, which shows how much better is the electric option than the conventional ones.

By taking into consideration the different restrictions a routing problem with electric vehicles and analyzing similar problems such as TSP the approach to solve this type of problem becomes clearer. This research paper builds on this by foundamenting its algorithmic base on these restrictions and on solutions to similar problems. Moreover, the mechanisms build upon said variables provides a near optimal way to tackle the vehicle routing problem with specific constraints.

Keywords

Algorithms, Routing, Vehicle Routing, Computer Science, Curriculum

ACM CLASSIFICATION Keywords

Theory of computation → Design and analysis of algorithms → Graph algorithms analysis → Shortest paths
Theory of computation → Design and analysis of algorithms → Approximation algorithms analysis → Routing and network design problems

1. INTRODUCTION

Humanity has always thrived for the improvement of its kind, searching for new ways to stay on top and progress as a species. The industrial revolution brought many changes to many societies, it transformed the world. It brought machines and industrialization to the working class. Everything became efficient and easily produced thanks to machines. One of the most affected systems was transportation. Before the fastest way to get somewhere by land was riding an animal, but after industrialization, came trains and later cars, that permitted a quicker mobilization. This gave massive growth and productivity across all the globe. But all this came at a price, oblivious to humans back then. Machines need fuel to work, and we quickly became dependant on it. At first it was carbon, but we quickly changed to fossil fuels after we learned how much energy they could provide to us. Of course, humans didn't know how harmful fossil fuels could be to the environment, and once we realized it was a bit too late. Many things nowadays derives or work from fossil fuels, especially our transportation. It is up to these newer generations to rectify this problem, to change and better the world instead of harming it. Electric vehicles provide an option to a more cleaner way to get around, avoiding the consumption of fossil fuels. Yet, this solution is still not perfect. As stated before, fossil fuels store a great amount of energy, and compared to the efficiency, we can take out of electricity for motors the cost and use of it is not worth it in many cases. By optimizing and finding the best possible routes and ways to mobilize electric vehicles the gap between these two can be lessen, giving the electric vehicles more viability in modern society.

2. PROBLEM

There is a need to find an algorithm that finds the most efficient route from one place to another for electric vehicles, taking into consideration the different conditions that affects them. By doing so, the use of electric vehicles will rise and possibly overshadow other types of powering, if its efficiency surpasses them. This will lead to a healthier environment and a better use of resources.

3. RELATED WORK

3.1 The traveling salesman agent.

The traveling salesman problem has been quite known for system engineers for a long time. It consist´s on a salesman who wants to visit different customers but doesn't know wich is the most efficient route to do so. The most famous solution to the problem is to use an heuristic approach. turning the clients into vertex, and the distance between them as the weight. After that, the algorithm used to determine the best route is the nearest neighbours (greedy algorithm). This way the traveler will always visit the closest vertex to the one he is standing in. This solution probes to be very efficient timewise, but has a margen error up to 25% of the optimal solution.

3.2 Finding the optimal route between two nodes in a graph

The question of how to get to one place to another has been asked for many centuries. With the long timed introduction of maps, the question has turned to "How is the fastest way I can get to one place to another". The best solution found so far for this problem has been Dijkstra´s algorithm, that also turns the map into a graph and analyses, for levels, the route that has the less weight between one vertex and another.

3.3 Delivery system problem.

A lot of delivery enterprises have to problem of choosing how many delivery trucks should be sent to cover the least amount of time. Using a single vehicle for every delivery has been proven to be non effective, so the need for an algorithm that balances the time vs cost efficiency arises. Similar to the traveling salesman, this problem utilices an heuristic approach to determine the route of every truck, this time including the parameters of cost for the deliveries, to check how many drivers should be sent and what route should they take.

4. ELECTRIC VEHICLE ROUTING ALGORITHM

4.1 Data Structure

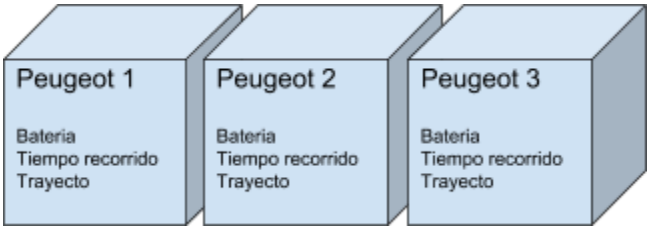

Figure 1: Array of the containers (data structure) labeled as 'Peugeot' which is a class which contains the battery level, time spent on delivery, and the current and past places it has been.

4.2 Operations of the Data Structure

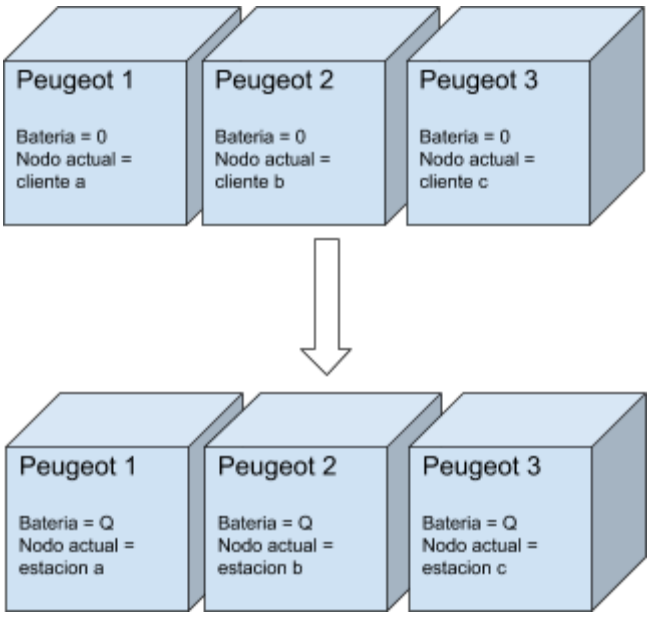

Figure 2: Charging of the peugeot fleet

4.3 Design criteria of the data structure

This data structure was chosen as the best data structure for the solution since it provides various advantages over other data structures. The container holds all relevant information for one car and its optimized for space efficiency since each value is assigned the least memory consuming type. The array, even though primitive and a bit space consuming, provides an extremely time efficient manner to solve the problem. This is due to the fact that it has a constant lookup time because of its indexing, letting a quick access to read and write on the container easily. Moreover, the array will pack all the containers more effectively, using less space than if stored with another data structure.

4.4 Complexity analysis

Method	Complexity
Mover Carro	O(1)
Recargar Carro	O(u)
Devolver a Deposito	O(m*u)

Table 1: Complexity analysis table for data structure where 'u' is the number of charging stations and 'm' the number of clients .

4.5 Execution time

	Datos 1	Datos 2	Datos n
Mover	182 ns	220 ns	170 ns
Recargar	220 ns	405 ns	284 ns
Dev. Dpo.	483279 ns	485709 ns	468384 ns

Table 2: Execution time of the different data structure methods in nanoseconds

4.6 Memory Used

	Datos 1	Datos 2	Datos N
Consumo de Memoria	12545 B	13874 B	12 KB

Table 3: Memory used by the data structure (Array of Peugeots) in Bytes.

4.7 Result Analysis

The results clearly show that the data structure is able to handle its methods and attributes with a decent space and time consumption. This proves that the chosen data structure is a good solution for the problem since it won't negatively affect the efficiency of any of the calculations used later on in the algorithm.

4.8 Algorithm

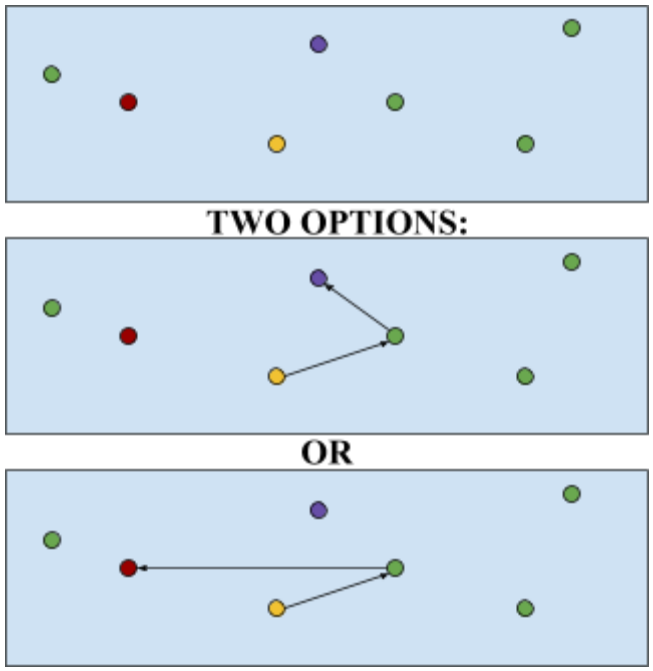

Figure 3: Calculation to decide which client to visit next. Yellow is a client node with a car on it, red is the deposit, green are unvisited clients, and purple is a charging station.

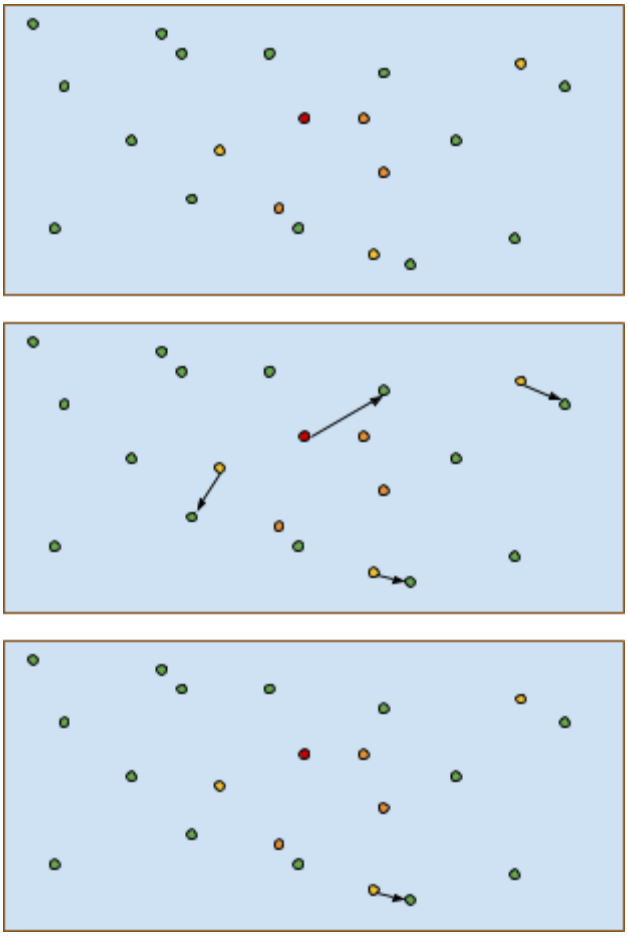

Figure 4: Example map showing the different client nodes and the deposit (red node). Green nodes are unvisited clients, the yellow ones are ones with delivery cars on them, the orange ones are clients that have already been visited. The algorithm calculated the nearest node from the deposit and from the yellow nodes using the restrictions on Figure 3, then it chooses the shortest distance(giving priority to the yellow nodes) to choose who to visit next.

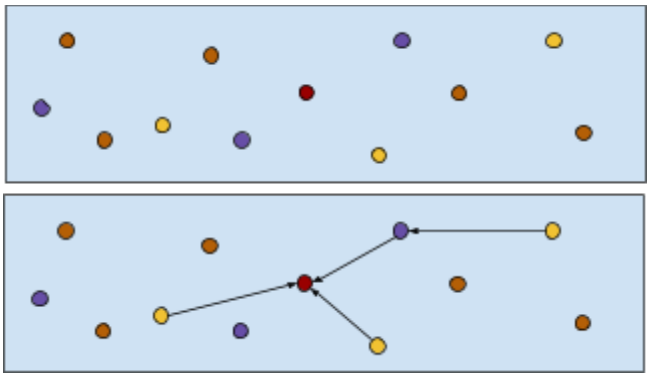

Figure 5: Once all the clients are visited the cars return to the deposit but charge first if they need to.

4.9 Complexity analysis of the algorithm

Subproblems	Complexity
Read data and create structures	$O(n^2)$
Next node from deposit	O(n)
Next node from cars	O(n.m.c)
Calculate routes	$O(n^2.m^2.u.c)$
Total complexity	$O(n^3.m^2.u^2.c)$

Table 4: Complexity of each subproblem of the algorithm. 'n' is the total number of nodes, 'm' the number of clients, 'c' the number of cars used in the fleet, and 'u' the number of charging stations. Note that the calculate routes subproblem requires solving the next node subproblems for cars and deposits and then takes into account the charging stations.

4.10 Design criteria of the algorithm

The algorithm was designed this way keeping in mind two key aspects, simplicity and efficiency. Since the problem presented a lot of restrictions the algorithm could not be overly complex since meeting all of this criteria is hard. By first taking into account all of the restriction variables the algorithm produces confidently solutions that meet all of the different restrictions it had. On a rudimentary level the algorithm is only calculating the shortest time (based on distance and charging times) to the next node, which is far from optimal, but by comparing these various shortest times between the deposit, the cars, and the unvisited clients, the algorithm can give a good approximation on what the routing of the vehicles should be. For the most part the algorithm is a linear one so the time and space consumption is also linear, but by using primitive, fast, and space efficient types and data structure is able to handle the problem quite well. Moreover, since the algorithm doesn't calculate every single possibility but instead takes a more logical approach its able to resolve the problem really quickly despite handling a large number of nodes (clients and charging stations).

4.11 Execution times

	DS 1	DS 2	DS N
Best case	890 ms	832 ms	890 ms
Average case	904 ms	901 ms	911 ms
Worst case	996 ms	1028 ms	998 ms

Table 5: Execution times of the algorithm to solve the problem with different data sets.

4.12 Memory consumption

	DS 1	DS 2	DS n
Memory consumption	10605 KB	10603 KB	10 MB

Table 6: Memory consumption of the algorithm with different data sets.

4.13 Analysis of the results

After extensively testing the algorithm it is clear that it works and is able to solve the problem with any of the data sets provided quickly. The results between each of the data sets was pretty consistent mainly for two reasons. First, the algorithm is pretty much a linear calculation since its based more on a logic approach than a extensive calculation. Second, the data sets were pretty similar in that they had a similar amount of data giving leading to similar results between each of the data sets. It is clear that the algorithm worked effectively and did so in an efficient manner.

5. CONCLUSIONS

This report shed light on some important factors about the electric vehicle routing problem. The problem had a lot of restrictions so deciding a data structure and designing an algorithm that could handle all of these restrictions was probably the most important part of the problem. I found out that keeping the solution effective and efficient through the decisions I made was hard since it was so linear, yet by making some logic decisions the algorithm was able to perform as expected.

Since I only used one data structure I was not able to compare it to others, yet a previous testing before the project to decide which one to use showed that the arrays, even though more space consuming than a HashMap or Tree was the fastest option that I could use, and wasn't the most space consuming one. It performed well.

5.1 Future Work

On the future I would like to follow the manual more closely, comparing my solutions with different alternatives to be clear which one is the most optimal one.

Regarding the solution I did I would maybe use more dynamic programing in order to safe space, but since it didn't show to be a problem maybe it's good as is.

REFERENCES

1. Adobe Acrobat Reader 7, Be sure that the references sections text is Ragged Right, Not Justified. http://www.adobe.com/products/acrobat/.

2. Fischer, G. and Nakakoji, K. Amplifying designers' creativity with domainoriented design environments. in Dartnall, T. ed. Artificial Intelligence and Creativity: An Interdisciplinary Approach, Kluwer Academic Publishers, Dordrecht, 1994, 343-364.